Statistical methods and genetic algorithms for prior knowledge selection.

Pierre Dangauthier, Anne Spalanzani, Pierre Bessiere

Méthodes statistiques et algorithmes génétiques pour la sélection de connaissances préalables.

Statistical methods and genetic algorithms for prior knowledge selection.

P. Dangauthier A. Spalanzani P. Bessière

CyberMove - INRIA Rhône-Alpes
655 Avenue de l'Europe, Montbonnot
38334 Saint Ismier cedex - France
pierre.dangauthier@laposte.net
{spalanzani,bessiere}@inrialpes.fr

Résumé

L'information perçue par un robot autonome plongé dans un environnement réel est abondante et incertaine. Il est alors difficile de traiter l'ensemble de cette information et d'en extraire la connaissance nécessaire à la réalisation d'une tâche donnée. Cet article présente différentes méthodes de sélection d'information permettant, dans le cas de la robotique, de trouver les corrélations existantes entre la tâche que le robot doit résoudre et les informations issues de ses capteurs. Nous montrons qu'à l'aide de ces méthodes, les temps de calcul et les taux de réussite pour la réalisation d'une tâche de reconnaissance sont améliorés. L'étude des capteurs sélectionnés nous permet de valider la pertinence de notre approche concernant la decouverte de nouvelles modalités sensori-motrices.

Mots Clef

Raisonnement en environnement incertain, robotique, algorithmes génétiques.

Abstract

Information perceived by a robot evolving in a real environment is abundant and uncertain. Therefore, it is difficult to deal with all this information and to extract information necessary for realizing a given task. This article presents different feature selection methods allowing, in a robotic framework, to find existing correlations between the task the robot has to perform and the information given by his sensors. We show that our methods enhance the computing time and the recognition rate computed by the robot. The study of selected sensors allows us to validate the relevance of our approach concerning the investigation for new sensoy-motor modalities.

Keywords

Reasoning in uncertain environment, robotic, genetic algorithms.

1 Introduction

Les systèmes sensori-moteurs qu'ils soient des animaux, des humains, des robots autonomes ou des personnages virtuels, sont destinés à évoluer dans un environnement non contrôlé. Cela signifie que les informations qu'ils possèdent sur leur environnement sont perpétuellement changeantes, incomplètes et parfois inconsistantes. Alors le modèle qu'ils construisent de leur environnement sera nécessairement incomplet.
Il apparaît alors qu'une approche basée sur une programmation déterministe classique pour créer des artefacts évoluant dans de tels environnements sera rapidement mise en échec. La solution proposée dans [3] est de transformer l'**incomplétude** du modèle en **incertitude** en modélisant le monde grâce aux **probabilités bayésiennes**. Il est possible de prendre des décisions motrices en utilisant la théorie des probabilités comme une extension de la logique. Cette approche est nommée : **programmation bayésienne des robots**([14]).

1.1 Contexte du projet

La programmation Bayésienne des robots. Dans toute la suite, nous utiliserons les conventions suivantes : $\forall i \in [0 \dots N-1]$ X_i est une variable qui peut prendre C valeurs $x_{i,k}$, $\forall k \in$

$[0 \dots C-1]$. On notera $P(X_i = x_{i,k})$ la probabilité de l'évènement $X_i = x_{i,k}$. Quand il n'y a pas d'ambiguïté, on la notera simplement $P(X_i)$.
Pour faire de la programmation bayésienne, et conformément à la vision bayésienne des probabilités, le programmeur commence par exprimer explicitement toutes ses connaissances *a priori*. C'est la phase de **description**, d'apport des **connaisssances préalables**.
Ensuite on pose une **question** qui est par exemple de la forme : "Quelle est la probabilité pour que $X_i = x_{i,k}$ sachant que $X_{j \neq i} = x_{j,k_j}$?".
Pour décrire le programme, on commence par spécifier la forme de la distribution de probabilité conjointe $P(X_0 \dots X_{N-1})$ en fonction de ses paramètres, puis on **identifie** ces paramètres par un apprentissage sur des exemples.
La **spécification** se décompose en : le **choix des variables pertinentes** $X_0 \dots X_{N-1}$, le choix d'une **décomposition** où le programmeur introduit ses *a priori* sur les indépendances conditionnelles entre variables, et le choix des **formes paramétriques** pour les distributions partielles.
Dans ce travail, on suppose que la décomposition est la suivante :

$$P(X_0 \dots X_{N-1},\, \theta) = P(\theta) \prod_{i=0}^{N-1} P(X_i|\theta)$$

Avec θ, le phénomène étudié et X_i les variables capteur.
Pour plus de détails sur les programmes bayésiens, on peut se référer à l'article [14] et à la thèse [18].

1.2 Plan de lecture

La section 2 décrit les méthodes existantes pour résoudre des problèmes similaires au nôtre. Nous verrons que les méthodes de sélection de caractéristiques sont diverses et variées. Cependant, nous verrons qu'il n'existe pas de cadre théorique unificateur ou du moins, que ces méthodes sont des approximations de ce que la théorie de l'information suggère.
Néanmoins, nous exhiberons un schéma général dans lequel s'inscrivent la majorité de ces méthodes et nous en distinguerons deux branches principales dont nous conserverons la dénomination anglophone : celle des *wrappers* et celle des *filters*.
Nous décrivons notre contribution dans la section 3 : la création et l'implémentation de plusieurs algorithmes de sélection de variables adaptés à notre problème. Ces méthodes se basent sur des algorithmes génétiques, des parcours heuristiques et la notion d'information mutuelle.
La section 4 décrit le simulateur logiciel et le robot qui nous ont permis de générer des fichiers de données de test. De plus, nous y évoquons notre façon de noter les algorithmes et de déterminer le taux de réussite de la classification.
Finalement, nous discuterons des résultats dans la dernière section.

2 Feature selection

La sélection de caractéristiques (*feature selection*) est un domaine très actif depuis quelques années, en particulier dans le cadre du *data mining*. En effet, la "fouille de données" dans de très grandes bases devient un enjeu crutial pour des applications telles que le génie génétique, la finance, les études de marché, les processus industriels complexes, etc.. Il s'agit en fait de résumer et d'extraire intelligement de la "connaissance" à partir de données brutes. La fouille de données est un domaine basé sur la statistique, l'appprentissage automatique et la théorie des bases de données.
La sélection de variables joue un rôle important dans le *data mining*, en particulier dans la préparation des données avant leur traitement. En effet, les intérêts de la sélection de variables sont les suivants :

- Lorsque le nombre de variables est vraiment trop grand (il peut aller jusqu'à plusieurs dizaines de milliers dans certaines applications), l'algorithme d'apprentissage ne peut pas terminer en un temps convenable. La sélection réduit la dimension de l'espace des caratéristiques.
- D'un point de vue intelligence artificielle, créer un classificateur revient à créer un modèle pour les données. Or, une attente légitime pour un modèle est d'être le plus simple possible (principe du Razoir d'Occam [2]). La réduction de la dimension de l'espace des caractéristiques permet alors de réduire le nombre de paramètres nécessaires à la description de ce modèle.
- Elle améliore la performance de la classification, sa vitesse et son pouvoir de généralisation.
- Elle augmente la compréhensibilité des données : on voit mieux quels sont les processus qui leur ont donné naissance.

Cette sélection consiste en :

- l'élimination de variables indépendantes de la classe,
- l'élimination de variables redondantes.

2.1 Principe général

Une structure générale des algorithmes de sélection de caractéristiques peut être proposée de la façon de la figure 1 ([16]). Jusqu'à ce qu'un certain critère soit satisfait, des sous-ensembles $\mathbf{Y}$ sont générés en parcourant l'espace des sous-

ensembles, et ils sont évalués. Nous décrivons successivement chaque phase.

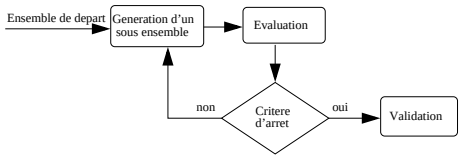

FIG. 1 – *Procédure générale de sélection d'un sous-ensemble de caratéristiques*

Génération de sous-ensembles. La génération de sous-ensembles est une procédure de recherche dans l'espace des sous-ensembles de cardinal 2^N. Tous les méthodes de parcours classiques sont utilisables. Par exemple [12] propose les méthodes de *forward addition* et *backward elimination*, [19] et [21] utilisent des algorithmes évolutionnistes.

2.2 Évaluation

L'évaluation d'un sous-ensemble est traitée de facons très diverses. Il existe deux grandes classes d'algorithmes :

1. Les *wrappers* : qui utilisent l'algorithme de classification pour évaluer les sous-ensembles générés,
2. Les *filters* : qui sont complètement indépendants de cet algorithme, mais se basent sur des considerations statistiques, entropiques, de cohérence, de distance,etc.

Les wrappers. Bien que conceptuellement plus simples que les filtres, les *wrappers* ont été introduits plus récemment par John, Kohavi et Pfleger en 1994 [10]. Leur principe est de générer des sous-ensembles candidats et de les évaluer grâce à l'algorithme de classification. Le score d'un ensemble sera par exemple un compromis entre le nombre de variables éliminées et le taux de réusite de la classification sur un fichier de test. Ainsi la phase "évaluation" du cycle de sélection est constituée par un appel à l'algorithme de classification. En fait, celui-ci est appellé plusieurs fois à chaque évaluation car un mécanisme de validation croisée (voir section 4.3) est fréquement utilisé.
De par son principe même, cette méthode génère des sous-ensembles bien adaptés à l'algorithme de classification. Les taux de reconnaissance sont hauts car la sélection prend en compte le biais intrinsèque de l'algorithme de classification. Un autre avantage est sa simplicité conceptuelle : il n'y a nul besoin de comprendre comment l'induction est affectée par la sélection de variables, il suffit de générer et de tester.

Cependant, trois raisons font que les *wrappers* ne constituent pas une solution parfaite. D'abord, ils n'apportent pas vraiment de justification théorique à la sélection et ils ne nous permettent pas comprendre les relations de dépendances conditionnelles (voir des compléments sur cette notion en annexe **??**) qu'il peut y avoir entre les variables. D'autre part la procédure de sélection est spécifique à un algorithme de classification particulier et les sous-ensembles trouvés ne sont pas forcement valides si on change de méthode d'induction. Finalement, et c'est le défaut principal de la méthode, les calculs deviennent vite très longs, voir irréalisables lorsque le nombre de variable croît.

Les filters. Les filtres n'ont pas les défauts des *wrappers*. Ils sont beaucoup plus rapides, ils reposent sur des considérations plus théoriques, ils nous permettent de mieux comprendre les relations de dépendance entre variables. Mais, comme ils ne prennent pas en compte les biais de l'algorithme de classification, les sous-ensembles de variables générés donnent un taux de reconnaissance plus faible.

Pour donner un score à un sous-ensemble, une première solution est de donner un score à chaque variable indépendemment des autres et de faire la somme de ces scores. Pour évaluer une variable, l'idée est de déterminer sa corrélation avec la variable de classe. Mais [9] propose des exemples simples montrant que cette approche nommée *feature ranking* pose des problèmes dans le cas général. En effet, cette approche n'élimine pas les variables redondantes, d'autre part il est possible que des variables peu corrélées avec la classe deviennent utiles lorsqu'on les considère dans le contexte des autres variables. Mais ce dernier reproche n'est pas valide dans le modèle capteur.

L'autre solution est d'évaluer un sous-ensemble dans sa globalité. On se raproche ici de l'apprentissage de la structure de réseau bayésien décrite dans [8]. Une méthode plus spécifique au problème est décrite dans [12]. Koller et Sahami y proposent d'éliminer une variable si elle possède une couverture markovienne, c'est-à-dire si elle est indépendante de la classe, sachant les autres variables. Ils proposent un algorithme utilisant une approximation sommaire de cette idée car elle n'est pas implémentable en pratique.

Il existe un intermédiaire entre *feature ranking* et *subset ranking* basé sur une idée de Ghiselli [5] et utilisé avec de bons résultats dans le cadre de la CFS (*correlation based feature selection*) par M. Hall [6]. Le score d'un sous-ensemble est construit en fonction des corrélations variable-

classe et des corrélations variable-variable selon la formule suivante :

$$r_{\theta S} = \frac{k\overline{r_{\theta i}}}{\sqrt{k + k(k-1)\overline{r_{ij}}}}$$

avec $r_{\theta S}$ le score du sous-ensemble de cardinal k, $\overline{r_{\theta i}}$ la moyenne arithmétique des corrélations entre θ et les variables i, et $\overline{r_{ij}}$ la moyenne des k^2 intercorrélations entre variables. Cette équation exprime que le score du sous-ensemble augmente si les variables sont fortement corrélées avec θ et diminue si elles sont fortement corrélées entre elles. De plus cette formule est valable dans le cadre de scores normalisés (*i.e.* de variance unitaire). L'idée est de dire qu'un bon sous-ensemble est constitué de variables hautement corrélées avec la classe (pour ne pas garder les indépendantes), et peu corrélées entre elles (pour éviter la redondance). Il s'agit d'une approximation car on ne prend en compte que les interactions d'ordre 1.

La corrélation ou dépendance entre deux variables peut être définie de plusieurs façons. Utiliser le coefficient de corrélation statistique comme dans [7] est trop restrictif car il ne capture que les dépendances linéaires. On peut, en revanche ,utiliser un test d'indépendance statistique comme le test du χ^2 [11][15][4]. Il est aussi possible d'utiliser la notion d'information mutuelle.

D'autres méthodes d'évaluation de sous-ensembles sont possibles. Par exemple, il est intéressant de se baser sur la vision bayésienne des probabilités pour construire des mesures d'indépendances strictement bayésiennes [17] [22] [20]. Certains auteurs [1] utilisent la notion de consistance pour juger un sous-ensemble. Des méthodes récentes combinant *wrapper* et *filter* sont présentées dans [9].

2.3 Critère d'arrêt

Le critère d'arrêt peut être de diverses natures : un temps de calcul, un nombre de générations (pour un algorithme génétique), un nombre de variables sélectionnées, une évaluation heuristique de la "valeur" du sous-ensemble...

2.4 Validation

Une fois le "meilleur" sous-ensemble déterminé, les performances de la classsification sont alors évaluées.

3 Algorithmes

Dans cette section, nous présentons 4 algorithmes de sélection de variables que nous avons définis, implémentés et comparés.

Notations : Les X_i sont les variables de capteurs, θ la variable de classe, **C=L** le nombre moyen de valeurs possibles pour toutes ces variables, **T** le nombre d'exemples et **k** le nombre de variables sélectionnées parmi les **N** initiales.

3.1 AGBN

Nom : AGBN = Algorithme Génétique, évaluation par apprentissage Bayésien Naïf.
Description :

1. Parcours de l'espace : algorithme génétique.
2. Évaluation des sous-ensembles : *wrapper*.
3. Critère d'arrêt : nombre de générations.

Un individu de l'algorithme génétique représente un sous-ensemble et la fonction d'évaluation retourne le taux de reconnaissance de l'algorithme de classification Bayésien naïf pour ce sous-ensemble.
Complexité :
Pour chaque individu de chaque génération, on effectue un apprentissage bayésien naïf sur un fichier de T données de taille $\overline{k} = \frac{N}{2}$ (en moyenne) et une classification sur un fichier similaire. Si **G** est le nombre de générations et **I** le nombre d'individus, le coût est :

$$O(GI(\overline{k}c^2 + TC\overline{k})) = O(GITC\overline{k})$$

opérations basiques (affectation, addition, multiplication...)

3.2 AGXH

Nom : AGXH = Algorithme Génétique, évaluation par l'entopie croisée (cross-H) .
Description :

1. Parcours de l'espace : algorithme génétique.
2. Évaluation des sous-ensembles : Formule de Ghiselli
3. Évaluation de corrélation : Entropie croisée. C'est une mesure de la "distance" entre deux distributions. L'entropie **H** d'une variable mesure l'incertitude pour cette variable. L'entropie croisée ou KL-distance (pour Kullback-Leibler [13]) entre deux distributions de probabilité μ et σ est :

   $$D_{KL}(\mu, \sigma) = \sum_x \mu(x) log \frac{\mu(x)}{\sigma(x)}$$

   On retrouve alors que l'information mutuelle entre deux variables X_i et X_j

   $$I(X_i, X_j) = \sum_{X_i} \sum_{X_j} P(X_i, X_j) log(\frac{P(X_i, X_j)}{P(X_i)P(X_j)})$$

est en fait la KL-distance entre les distributions $P(X_i, X_j)$ et $P(X_i)P(X_j)$. En effet, elle est nulle si les variables sont indépendantes et d'autant plus grande que les variables sont dépendantes.

Comme cette information mutuelle est biasée en faveur des variables avec un grand nombre de valeurs possibles, nous utilisons le coefficient d'incertitude symétrique CIS défini par :

$$CIS = 2[\frac{I(X_i, X_j)}{H(X_i) + H(X_j)}]$$

Le CIS vaut 0 si les variables sont indépendantes (l'information mutuelle est nulle) et 1 si $H(X_i|X_j) = 0$, c'est-à-dire si l'incertitude sur X_i connaissant X_j est nulle, *i.e.* si elles sont parfaitement liées.

4. Critère d'arrêt : nombre de générations.

Complexité :

A première vue $O(GI\overline{k}^2C^2)$, mais $O(GI\overline{k}^2)$ grace à nos optimisations.

3.3 FAXH

Nom : FAXH = *Forward Addition*, évaluation par l'entopie croisée (XH).

Description :

1. Parcours de l'espace : *Forward Addition*.
2. Évaluation des sous-ensembles : Formule de Ghiselli.
3. Évaluation de corrélation : Entropie croisée.
4. Critère d'arrêt : On s'arrête quand il n'est plus possible de faire croître le score en ajoutant une variable.

C'est la même méthode que AGXH pour évaluer les sous-ensembles, mais le parcours de l'espace est différent. On part de l'ensemble vide et on ajoute les variables une à une. Soit S le sous-ensemble courant. Pour chaque variable $X_i \notin S$, on évalue tous les $S_i = S \bigcup X_i$. On retiendra le sous-ensemble de plus grand score.

Complexité : $O(\sum_{k=0}^{N}(N - k)kC^2) = O(N^3C^2)$

3.4 ModCapt

Nom : ModCapt car c'est le seul à prendre en compte explicitement l'hypothèse de MODèle CAPTeur.

Description

1. Parcours de l'espace : *backward elimination*
2. Évaluation des sous-ensembles : non défini,
3. Évaluation de corrélation : non défini,
4. Critère d'arrêt : on arrête quand on a éliminé un nombre prédéfini de variables ($\frac{N}{2}$ dans nos tests).

On suppose l'hypothèse du modèle capteur. On part de l'ensemble plein, et on enlève des variables une à une. A chaque essai, on quantifie la distance entre la densité de probabilité originelle et la densité en considérant la variable indépendante des autres conditionnellement à la classe. La distance utilisée est la KL-distance symétrisée Δ.

$$\Delta(\mu, \sigma) = D_{KL}(\mu, \sigma) + D_{KL}(\sigma, \mu)$$

On éliminera les $\frac{N}{2}$ variables qui rendent cette distance minimale. Soit P la distribution sur N variables en supposant le modèle capteur et P_i la même distribution en supposant de plus que X_i est indépendante de la classe θ. On a :

$$\begin{aligned} P(X_0 \dots X_{N-1}\theta) &= P(\theta) \prod_{k=0}^{N-1} P(X_k|\theta) \\ P_i(X_0 \dots X_{N-1}\theta) &= P(\theta)P(X_i) \prod_{k=0\ k\neq i}^{N-1} P(X_k|\theta) \end{aligned}$$

et alors :

$$\begin{aligned} \Delta(P, P_i) = & \sum_{\theta} P(\theta) \sum_{X_i} log(\frac{P(X_i)}{P(X_i|\theta)}) \\ & -(P(X_i) - P(X_i|\theta)) \end{aligned}$$

Complexité : $O(NC^2)$

Remarques

On a ainsi retrouvé que, dans le cadre du modèle capteur, il n'est pas nécessaire de considérer les autres variables pour décider si X_i est indépendante de θ. On retrouve qu'il suffit de regarder "l'écart" entre $P(X_i)$ et $P(X_i|\theta)$. On pourrait alors penser qu'il suffit de classer les variables selon leur corrélation avec θ et de prendre les meilleures (*feature ranking*). Mais, comme nous l'avons vu dans la section 2.2, cette approche n'élimine pas les variables redondantes. L'indépendance conditionnelle n'a rien à voir avec la redondance. Par exemple si on condidère deux fois la même variable on a $P(X|\theta, X) = P(X|\theta)$ alors que X est évidemment redondante avec elle-même.

On remarque d'autre part que l'hypothèse du modèle capteur permet de passer d'une complexité astronomique en $O(C^{N+1})$ à une complexité linéaire en N.

4 Évaluation

Dans cette section, nous présentons notre approche d'évaluation et de validation de nos algorithmes. Pour cela nous décrivons nos méthodes

de génération de fichiers de données , puis nous présentons notre estimateur du taux de réussite de la classification.

4.1 Génération de données

Nous avons deux sources de données : un simulateur logiciel et un vrai robot.

Simulateur. Dans un premier temps, il nous a paru nécessaire de disposer de données artificielles complètement maitrisées. En effet cela nous a permis de mieux comprendre les relations de dépendances entre les variables pour mieux étudier les réponses de nos algorithmes. De plus, il est très commode de pouvoir générer les données qui nous intéressent *a priori*, afin de tester un aspect particulier d'une méthode. Ainsi, nous avons codé une plateforme expérimentale simulant les sorties des capteurs d'un robot stimulés par différentes sources.
Le simulateur modélise un ou plusieurs robots dans un univers en 2D . Dans cet univers, divers objets peuvent se mouvoir selon des trajectoires précalculées ou définies par l'utilisateur. Les objets de l'environnement sont perçus par le robot par l'intermédiaire de ses capteurs. Un robot peut posséder un nombre arbitraire de capteurs. Les capteurs peuvent être sensibles à différentes grandeures (lumière, son, proximité ...).
Ce simulateur nous permet de générer de nombreux fichiers de données, chacun testant un aspect particulier des algorithmes. De plus, nous sommes maîtres du nombre de variables, et donc de la complexité des calculs à effectuer.

Robot. La deuxième phase de l'évaluation est réalisée sur des données robotiques réelles provenant de plusieurs expériences. Le robot utilisé est un **Koala** commercialisé par la société **K-Team S.A.**. Il s'agit d'un robot mobile de taille moyenne (70 cm par 40 cm) monté sur 6 roues. Il est équipé de nombreux capteurs dont 16 proximètres à ultrasons, 16 photomètres, des odomètres et une caméra vidéo mobile selon deux axes.
L'expérience réalisée est celle du *suivi de balle rouge*. Le Koala est immobile pendant toute l'expérience. grâce à sa caméra il peut reconnaître des objets colorés. Ce suiveur (*tracker*) est une implémentation classique de la règle de Bayes et se base sur la couleur. La caméra est asservie à fixer le centre de l'objet mobile. Nous choisissons de ne déplacer l'objet (la balle rouge) que dans le plan horizontal. Tout ceci nous donne accès à une image de l'angle entre l'axe pricipal du robot et la balle rouge. Durant l'expérience, on déplace la balle autour du robot en prenant soin que sa présence stimule les proximètres. Toutes les 100ms, on recueille 59 valeurs sur le robot.
Ces 58 (59 - l'angle θ) variables seront soumises à nos algorithmes de sélection. Nous cherchons quelles sont les variables parmis elles qui sont nécessaires et suffisantes pour prévoir la position de la balle rouge (ou d'un autre objet) sans utiliser la caméra. Le but est de trouver que seuls les proximètres sont porteurs d'information sur la position de la balle.

4.2 Plan d'expériences

Pour valider et comparer les méthodes implémentées, un protocole de test a été mis en place. Il commence avec des expériences très simples avec le simulateur et finit par l'utilisation de données robotiques. Les premiêres expériences sont très simples (peu de variables) et nous permettent de vérifier si nos algorithmes sont capables de détecter des redondances ou des indépendances triviales. Nous produisons ensuite des fichiers de tests plus réalistes (plus de 20 variables) et des fichiers provenant du robot afin de valider nos méthodes sur des données du monde réel.

4.3 Validation

Dans la littérature, la sélection de caractéristiques est souvent un prétraitement pour faire de la classification. Les auteurs valident donc leurs sous-ensemble de variables par les performances du programme d'apprentissage utilisé ensuite. En général, ils comparent les taux de réussite de la classification, avant et après la sélection.
Dans notre cas, cette démarche est tout à fait pertinente car nous voulons aussi prévoir la valeur d'une variable avec moins d'informations. De plus, dans le contexte de la programmation bayésienne des robots, le choix de la méthode d'apprentissage à été très naturel : on utilise un apprentissage bayésien.
Comme nous l'avons vu dans l'introduction, nous supposons l'indépendance des variables conditionnellement à la classe. Ceci nous amène à utiliser un apprentissage bayésien naïf.
Comme nous avons à notre disposition un assez grand nombre de données (le simulateur en produit à la demande et le robot en génère dix par seconde), nous avons choisit une méthode de **validation simple**. On apprend sur un fichier de données et on mesure le taux de reconnaissance sur un autre fichier, similaire mais différent. Dans le cas du simulateur le fichier de test correspond au même mouvement de la même source, mais il diffère de par le bruit des capteurs qui n'est pas le même (en fait le générateur de hasard a été initialisé différemment). Ceci nous rapproche des

conditions réelles dans lesquelles le robot fait face à de nouveaux stimulis. Pour le cas des données robotiques, on a validé avec le même fichier et avec un fichier relevé lors d'une autre expérience.

5 Résultats

Pour interpréter les résultats et comparer nos algorithmes, nous nous sommes basés sur les critères suivants :

- Nombre de variables retenues,
- Taux de reconnaissance sur un fichier similaire mais différent de celui ayant servi à l'apprentissage,
- Temps de calcul sur PC 400 MHz,
- Pouvoir explicatif du sous-ensemble, capacité de généralisation (sont-ce les "bonnes" variables ?),
- Nombre de paramètres à fixer à la main.

Nous commencerons par étudier les résultats bruts des expérimentations, puis nous nous lancerons dans des analyses de plus en plus fines des résultats.

5.1 Première discussion

On analyse rapidement les résultats afin d'éliminer tout de suite les mauvais algorithmes.

Moyenne sur tous les fichiers. Le tableau 1 présente une moyenne sur les 11 fichiers de tests des résultats bruts. La dernière colonne (**ALL**) donne le nombre total de variables du fichier d'exemples, le taux de réussite de la classification en éliminant aucune de ces variables et le temps de calcul de la classification. Les résultats ont été exprimés par rapport à cette dernière colonne. Par exemple, l'algorithme AGBN a, en moyenne, gardé 62% des variables tout en améliorant faiblement le taux de réussite de la classification d'un facteur 1,04 (104 % du taux originel avec ALL). On voit par contre qu'il est très couteux de l'utiliser ($\frac{4586}{51} = 90$ fois plus lent que AGXH). Remarque : les moyennes ont été arrondies à l'entier le plus proche.

Moyenne sur les "gros" fichiers. Le tableau 2 à été constuit sur le même principe que le tableau 1 sauf qu'on n'a considéré que les fichiers d'expériences réalistes, *i.e.* les expériences pour lesquelles on avait plus de 20 variables.

On peut d'emblée mettre de côté le *wrappers* qui est vraiment trop lent pour de la robotique autonome. Dans l'expérience robotique AGBN a mis près d'une heure. Ceci nous pousse à le rejeter car ce calcul est beaucoup trop lourd pour un processeur embarqué. Cependant, dans le cadre d'une autre application *off line*, il aurait été pertinent de le garder car, comme on l'attendait il maximise le taux de bonnes classifications.

Comme l'estimation de complexité l'avait prévu ($(O(N^3)$), FAXH est assez lent. Ceci est dû à sa méthode de parcours de l'espace des sous-ensembles qui est très répétitive. Elle pourrait être optimisée en rendant certains calculs incémentaux.

AGXH et ModCapt. À ce stade, nos meilleurs candidats sont AGXH et ModCap. ModCap donne un meilleur taux, mais il garde plus de variables. Cependant il faut rappeller que le critère d'arrêt de cet algorithme est justement de sélectionner $\frac{N}{2}$ variables. Pour mieux le comparer on a refait des tests en changeant ce critère d'arrêt par $\frac{N}{4}$ variables. Les résultats sont les suivants, sur les "gros" fichiers seulement :

On contate alors qu'avec un nombre de variables retenues équivalent , ModCap reste plus performant que AGXH, et beaucoup plus rapide. Cet algorithme semble donc le meilleur, si l'on considère qu'on sait *a priori* le nombre de variables à garder. Or, ce n'est pas du tout le cas dans le cadre de notre application robotique dans lequel on veut obtenir un maximum d'autonomie.

Etude de MoCap. A part le problème du critère d'arrêt, l'algorithme MoCap s'est bien comporté. Il est rapide et les variables sélectionnées sont les bonnes dans le sens où elles entrainent un taux de reconnaissance souvent supérieur à celui donné par l'ensemble des N variables. Il conserve en général les bonnes variables. On voit d'ailleurs dans cette expérience que les photomètres sont plus corrélés avec θ que les proximètres. AGXH les a aussi retenus avec en plus un des proximètres. Dans les "grosses" expériences, on constate que MoCap est moins performant qu'AGXH dans l'élimination des variables redondantes, mais qu'il est sait mieux débusquer les variables indépendantes de θ.

5.2 Comparatif

Après une étude détaillée des sous-ensembles générés, nous pouvons résumer nos conclusions dans le tableau 4.

5.3 Retour sur le changement de modalité

En regardant de plus près les résultats de l'expérience de la balle rouge, on constate d'abord que sur les 58 variables initiales le test du chi^2 et AGBN ont sélectionné presque toutes les variables non constantes. Ce défaut pourrait être corrigé en modifiant leurs paramètres (baisser les

seuil de signification du test et modifier la fonction d'évaluation de l'algorithme génétique).
D'autre part on lit sur le tableau 5 que les taux de reconnaissance sont faibles, même avec toutes les variables. Ceci est du à l'expérience elle même et non au méthodes de sélection. Etant donné que, lors de l'expérience, la balle ne faisait pas le tour entier du robot, tous les proximètres n'ont pas été stimulés.
On constate alors que AGXH, FAXH et ModCap$\frac{N}{4}$ ont des performances équivalentes. Mais comme les deux premiers sont plus lents et comme il faut dire explicitement à ModCap de garder $\frac{N}{4}$ variables, nous considérons que AGXH est le plus adapté à la découverte d'un nouvelle modalité sensorielle pour cette expérience.

6 Conclusion

6.1 Contribution

Dans le but de permettre la découverte de nouvelles modalités sensori-motrices, nous avons comparé plusieurs méthodes de sélection de variables. Pour cela nous avons testé différents algorithmes selon des critères objectifs de performances et de coût. Nous en avons conclu que bien que le choix final dépende de l'application visée, deux algorithmes sortent du lot : MoCapt et AGXH. Tous deux sont basés sur une mesure entropique, mais il diffèrent de par leur méthode de parcours de l'espace des sous-ensembles de variables.
Ces methodes, utilisées dans le cadre de la robotique permettent de s'affranchir d'une part de l'information perçue par les capteurs du robot. Ceci nous permet non seulement de réduire les temps de réaction du robot, mais également d'améliorer la robustesse de son comportement. Ce qui nous encourage à poursuivre dans la découverte automatique de nouvelles modalités sensori-motrices permettant au robot de s'adapter aux aléats non prévisibles de son environnement.

6.2 Perspectives

A un plus haut niveau d'abstraction, nous n'avons réalisé qu'un premier pas vers la découverte entièrement automatique de nouveaux comportements. Des questions ouvertes sont : comment le robot indentifiera-t-il les périodes pendant lesquelles il doit tenter une telle découverte ? Doit-il le chercher en permanence et en parallèle avec ses autre tâches ?
D'autre part, nous ne nous sommes intéressé qu'au variables sensorielles, mais il est tout aussi pertinent de rechercher des corrélations dans le domaine moteur. Quels seraient les enjeux d'une telle approche ?

%	AGBN	AGXH	FAXH	ModCap	ALL
NbVar	62	34	35	45	100
TxReco	105	83	101	98	100
TpsCalc	4586	51	927	4	1

TAB. 1 – *Moyennes des résultats. Les valeurs sont exprimées relativement à la dernière colonne qui présente* **k=N**, **Tx** *et* **t** *pour l'ensemble de toutes les variables.*

%	AGBN	AGXH	FAXH	ModCap	ALL
NbVar	52	25	25	41	100
TxReco	106	87	102	103	100
TpsCalc	2273	23	401	1	1

TAB. 2 – *Moyennes des résultats sur les fichiers réalistes.*

%	AGXH	ModCap$\frac{N}{2}$	ModCap$\frac{N}{4}$
NbVar	25	41	23
TxReco	87	103	99
TpsCalc	23	1	1

TAB. 3 – *Moyennes des résultats sur les "gros" fichiers pour AGXH et ModCap avec comme critères d'arrêts* $\frac{N}{2}$, $\frac{N}{4}$ *variables.*

Critère	AGBN	AGXH	FAXH	ModCap
Temps	—	+	—	+++
TxReco	++++	+	++	+++
Nombre variables	–	+	+	paramètre
Explication	–	+	+	++
Paramètres	-	-	++	—

TAB. 4 – *Forces et faiblesses des différents algorithmes. Le nombre de variables retenues par ModCap est l'un de ses paramètres. La ligne "explication" exprime si l'algorithme a bien sélectionner les variables "logiques", par exemple des photomètres si la source est lumineuse. La ligne "paramètres" exprime la difficulté à trouver les bons paramètres de l'algorithme.*

	AGXH	FAXH	ModCap$\frac{N}{4}$	AGBN	ALL
Proximètres retenus (sur 16)	6	7	6	13	16
Autres cap. retenus (sur 42)	9	8	8	22	42
TxReco	0.8	0.8	0 .76	0.91	0.89

TAB. 5 – *Nombre de proximètres, d'autres capteurs retenus et taux de reconnaissance lors de l'expérience de la balle rouge.*

Références

[1] Hussein Almuallim and Thomas G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2) :279–305, 1994.

[2] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6) :377–380, 1987.

[3] Bessière et al. Interprétation versus description (i) : Proposition pour une théorie probabiliste des systèmes cognitifs sensori-moteurs. *Intellectica*, 1999.

[4] Olivier Gaudoin. Méthodes statistiques pour l'ingénieur. grenoble, france, 2002.

[5] Edwin E. Ghiselli. *Theory of Psychological Measurement*. McGraw-Hill Book Company, 1964.

[6] M. Hall. Correlation-based feature selection for machine learning, 1998.

[7] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 359–366. Morgan Kaufmann, San Francisco, CA, 2000.

[8] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks : The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[9] André Elisseeff Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 2003.

[10] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http ://citeseer.nj.nec.com/13663.html.

[11] Sir Maurice Kendall and Alan Stewart. *The Advanced Theory of Statistics, Volume 1, 4th Edition*. Mcmillan Publishing, New York, 1977.

[12] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[13] S. Kullback and Leibler. On information and suoeciency. page 22 :7986, 1951.

[14] Diard J. Lebeltel O., Bessière P. and Mazer E. Bayesian robots programming. *Autonomous Robot 2003 (in press)*, 2003.

[15] H. Liu and R. Setiono. Chi2 : Feature selection and discretization of numeric attributes, 1995.

[16] H. Liu and L. Yu. Feature selection for data mining, 2002.

[17] Dimitris Margartitis and Sebastian Thrun. A bayesian multiresolution independence test for continuous variables. In *Uncertainty in Artificial Intelligence : Proceedings of the Seventeenth Conference (UAI-2001)*, pages 346–353, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[18] Lebeltel O. Programmation bayésienne des robots. thèse de l'institut national polytechnique de grenoble, france, 1999.

[19] Oliver Ritthoff and et al. A hybrid approach to feature selection and generation using an ea, 2002.

[20] David R. Wolf. Mutual information as a bayesian measure of independence, 1994.

[21] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13 :44–49, 1998.

[22] Marco Zaffalon and Marcus Hutter. Robust feature selection by mutual information distributions, 2002.